ORACLE CONFIDENTIAL

"Express Mail" Mailing Label No. **EL960828301US**

PATENT APPLICATION
ATTORNEY DOCKET NO. OR03-15501

UNITED STATES UTILITY PATENT APPLICATION

FOR

METHOD AND APPARATUS FOR PROVIDING QUERY-LEVEL SECURITY IN A DATABASE SYSTEM

INVENTOR:

Daniel ManHung Wong

Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065

METHOD AND APPARATUS FOR PROVIDING QUERY-LEVEL SECURITY IN A DATABASE SYSTEM

Inventor: Daniel ManHung Wong

BACKGROUND

Field of the Invention

[0001] The present invention relates to database security. More specifically, the present invention relates to a method and an apparatus for providing query-level security for a database system.

Related Art

[0002] Databases commonly store highly sensitive data, such as salaries, corporate financial data, and even classified military secrets. Consequently, database systems are typically designed to prevent unauthorized accesses to sensitive data. This problem is compounded by the fact that middle-tier applications often access a database on behalf of various users. Consequently, the database system must often rely on applications to provide access control mechanisms. Although applications that access databases typically ensure that a given query originates from an authorized user, many of these applications are vulnerable to a form of attack known as "SQL injection."

[0003] In order to perform SQL injection, a user provides an input to an application which includes an SQL statement. In doing so, the user knows that the application will incorporate this input, which includes the SQL statement, into a query, and that the SQL statement will cause the query to retrieve data which is different from the data that the application intended to retrieve.

[0004] For example, suppose a user enters the value "5" into a ProdID field of a web form, so that the web form submits the value 5 to an associated application. The application then forms a SQL statement such as:

SELECT prize, color FROM inventory WHERE ProdID = 5.

This query returns the values from the prize and color columns of the inventory table for an entry where the ProdID field contains the value 5.

[0005] Instead of simply entering the value "5" into the web form, the user can input the string "5 OR 1=1" into the web form. When the application substitutes this string into the query, the following query is formed.

SELECT prize, color FROM inventory WHERE ProdID = 5 OR 1=1.

This query, in contrast to the original query, returns values from the prize and color columns in *every* row of the inventory table!

[0006] The above is just one example of SQL injection. This is a well-known problem and will not be described further.

[0007] Currently, database applications perform tests on queries to detect SQL injection. While these tests can be effective, there are many drawbacks to this solution. Requiring each application to perform tests for SQL injection requires a significant amount of effort on the part of the application developers to include code into applications to perform these tests. Additionally, since different applications are typically developed by different developers, there is generally a lack of consistency in applying these tests across different applications.

[0008] Hence, what is needed is a method and an apparatus for providing query-level security for a database without the problems described above.

SUMMARY

[0009] One embodiment of the present invention provides a system that facilitates using query signatures to provide security for a database system. During operation, the database system receives a query. Next, the system parses the query to determine a signature for the query. This signature specifies a structure based on operators for the query and is independent of the value of literals in the query. The system then determines if the signature can be found in a signature cache which contains valid query signatures. If so, the system processes the query.

[0010] In a variation of this embodiment, if the signature is not in the signature cache, the system triggers a mismatch alert

[0011] In a further variation, the mismatch alert throws an error.

[0012] In a further variation, the mismatch alert is sent to a database administrator and the query is processed.

[0013] In a further variation, the mismatch alert is sent to a requesting application, thereby allowing the requesting application to take action.

[0014] In a further variation, the signature cache is initialized by recording signatures of valid transactions during a system initialization operation.

[0015] In a further variation, if the signature generates a mismatch alert and if the query is a valid query, the system allows a database administrator to add the signature to the signature cache.

BRIEF DESCRIPTION OF THE FIGURES

[0016] FIG. 1 illustrates a database system in accordance with an embodiment of the present invention.

[0017] FIG. 2 illustrates a client in accordance with an embodiment of the present invention.

[0018] FIG. 3 illustrates an application server in accordance with an embodiment of the present invention.

[0019] FIG. 4 illustrates a database server in accordance with an embodiment of the present invention.

[0020] FIG. 5 presents a flowchart illustrating the process of initializing a signature cache in accordance with an embodiment of the present invention.

[0021] FIG. 6 presents a flowchart illustrating the process of validating a query in accordance with an embodiment of the present invention.

[0022] FIG. 7 presents a flowchart illustrating the process of responding to a mismatch alert in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION

[0023] The following description is presented to enable any person skilled in the art to make and use the invention, and is provided in the context of a particular application and its requirements. Various modifications to the disclosed embodiments will be readily apparent to those skilled in the art, and the general principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the present invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein.

[0024] The data structures and code described in this detailed description are typically stored on a computer readable storage medium, which may be any device or medium that can store code and/or data for use by a computer system. This includes, but is not limited to, magnetic and optical storage devices such as disk drives, magnetic tape, CDs (compact discs) and DVDs (digital versatile discs or digital video discs), and computer instruction signals embodied in a transmission medium (with or without a carrier wave upon which the signals are modulated). For example, the transmission medium may include a communications network, such as the Internet.

Database System

[0025] FIG. 1 illustrates a database system 100 in accordance with an embodiment of the present invention. Database system 100 includes client 102, application server 106, database server 110, and database 112. Client 102 is coupled to application server 106 across network 104, while database server 110 is coupled to application server 106 across network 108. Database 112 is coupled to database server 110.

[0026] Networks 104 and 108 can generally include any type of wire or wireless communication channel capable of coupling together computing nodes. This includes, but is not limited to, a local area network, a wide area network, or a combination of networks. In one embodiment of the present invention, networks 104 and 108 include the Internet. Network 108 can also be a private network. Note that in some configurations application server 106 and database server 110 can be hosted by the same computer system.

[0027] Database 112 can include any type of system for storing data in non-volatile storage. This includes, but is not limited to, systems based upon magnetic, optical, and magneto-optical storage devices, as well as storage devices based on flash memory and/or battery-backed up memory.

[0028] Client 102 allows a user (not shown) to enter data through a web browser. This data is sent to application server 106 across network 104. Application server 106 then forms a SQL query using the data supplied by client 102 and forwards this SQL query to database server 110 across network 108.

[0029] Database server 110 validates the query and, if valid, performs the requested operation on database 112. In doing so, the system returns any requested data to application server 106. If the query is not valid, database server 110 triggers a mismatch alert.

Client

[0030] FIG. 2 illustrates client 102 in accordance with an embodiment of the present invention. Client 102 includes browser 202 and network interface 204. Browser 202 can generally include any type of web browser capable of viewing a web site, such as the INTERNET EXPLORER^{TM} browser distributed by the Microsoft Corporation of Redmond, Washington. A user enters data for an application running on application server 106 through browser 202. This data is sent to the application running on application server 106 across network 104 by network interface 204. Network interface 204 also receives data from the application running on application server 106 and provides this data to browser 202 so that browser 202 can display the data to the user.

Application Server

[0031] FIG. 3 illustrates application server 106 in accordance with an embodiment of the present invention. Application server 106 includes browser interface 302, application 304, and database server interface 306. Browser interface 302 includes a web site that communicates with browser 202 across network 104. This web site provides web pages to be presented to a user by browser 202 and receives data entered by the user. The data entered by the user is forwarded to application 304.

[0032] Application 304 incorporates the data received from the user into SQL queries and forwards these queries to database server 110 through database server interface 306. In the other direction, database server interface 306 receives data from database server 110 and forwards this data to application 304, which forwards the data to client 102.

Database Server

[0033] FIG. 4 illustrates a database server 110 in accordance with an embodiment of the present invention. Database server 110 contains application interface 402, signature cache initializer 404, signature cache 406, signature cache comparator 408, and database interface 410.

[0034] Application interface 402 communicates with applications on application server 106 and receives queries from the applications and supplies data to the applications. Queries received from application server 106 are sent to signature cache comparator 408 for validation.

[0035] Signature cache initializer 404 initializes signature cache 406 with valid signatures for allowed queries. During an initialization operation, applications on application server 106 are exercised to generate allowed queries in a controlled environment. Signature cache initializer 404 parses these queries to determine a signature for each query. These signatures are stored in signature cache 406. Note that this initialization operation can be part of a regression test prior to release of the system.

[0036] A signature for a query specifies a structure for the query base on operations within the query. The signature includes the keywords for the query but remains independent of the literals within the query. This allows the system to validate a query based upon its signature and to reject an invalid query based upon its signature. This operation is described in more detail below.

[0037] Signature cache 406 includes the valid signatures that an application can use when querying the database. After signature cache 406 has been initialized, the data in signature cache 406 can be saved to and reloaded from the database. Any acceptable lookup structure can be used for signature cache 406, for example, a hash table or linked list can be used. Note that signature caches already exist in databases and are used to facilitate query processing.

[0038] The signature for a SQL query can be the keywords of the SQL query with the literals removed from the SQL query. For example, the signature for the SQL query

SELECT prize, color FROM inventory WHERE ProdID = 5

can be

SELECT FROM WHERE =.

[0039] If this signature is stored in the signature cache, the SQL query is allowed to proceed. However, if the SQL query has been modified by SQL injection to be

SELECT prize, color FROM inventory WHERE ProdID = 5 OR 1=1, the signature for the query is

SELECT FROM WHERE = OR =.

[0040] Since this signature is not in the signature cache, the query is not allowed to proceed.

Initializing the Signature Cache

[0041] FIG. 5 presents a flowchart illustrating the process of initializing a signature cache in accordance with an embodiment of the present invention. The system starts by executing the applications in a controlled environment, possibly during regression testing of the applications. During the testing process, the system traps all database queries (step 502). Next, the system parses the database queries to produce a set of valid signatures (step 504).

[0042] The system then saves the valid signatures in the signature cache (step 506). Finally, the system stores the signature cache, possibly in the database, for future recall (step 508).

Validating a Query

[0043] FIG. 6 presents a flowchart illustrating the process of validating a query in accordance with an embodiment of the present invention. The system starts by retrieving the valid signature cache from its storage location (step 602). Next, the system receives a database query from the application (step 604). The system then parses the query to determine its signature (step 606).

[0044] After determining the signature for the query, the system compares the signature to valid signatures in the signature cache (step 608). The system then determines if a match was found (step 610). If so, the system processes the query (step 612). If not, the system triggers a mismatch alert (step 614). After processing the query at step 612 or triggering a mismatch alert at step 614, the system returns to step 604 to process a subsequent query.

Mismatch Alert

[0045] FIG. 7 presents a flowchart illustrating the process of responding to a mismatch alert in accordance with an embodiment of the present invention. The system starts when a mismatch alert is received from the query validation portion of the system (step 702). After receiving the mismatch alert, the system determines how the handler is set to process the mismatch alert (step 704). Note that there are many possible techniques to respond to a mismatch alert. Three possible techniques are described herein.

[0046] If the handler is set to an error mode, the system processes the error (step 706). This can include informing an administrator or informing the application of the error. If the handler is set to a continue mode at step 704, the system first notifies an administrator of the mismatch (step 708). After informing the administrator of the mismatch, the system processes the query (step 710). If the handler is set to a notify application mode, the system notifies the application, thereby allowing the application to determine the proper course of action (step 712). The process is complete after steps 706, 710, or 712.

[0047] The foregoing descriptions of embodiments of the present invention have been presented for purposes of illustration and description only. They are not intended to be exhaustive or to limit the present invention to the forms disclosed. Accordingly, many modifications and variations will be apparent to practitioners skilled in the art. Additionally, the above disclosure is not intended to limit the present invention. The scope of the present invention is defined by the appended claims.